SYSTEM AND METHOD FOR DESIGNING AND OPTIMIZING THE MEMORY OF AN EMBEDDED PROCESSING SYSTEM

TECHNICAL FIELD OF THE INVENTION

The present invention is generally directed to memory design applications and, more specifically, to a system design tool capable of determining the memory access usage of an embedded processing system and selecting and optimizing the types and sizes of memories used in the embedded processing system.

BACKGROUND OF THE INVENTION

The demand for high-performance processing devices requires that state-of-the-art integrated circuits perform operations in the minimum amount of time, consume the minimum amount of power, and occupy the smallest amount of die space possible. This is particularly true of a wide array of embedded processing systems, such as application-specific integrated circuit (ASIC) devices, that contain a processor and memory. ASIC devices and other embedded processing systems are used in network cards, modems, wireless receivers and transmitters, smart cards, cell phones, personal digital assistant (PDA) devices, and the like.

In normal software development efforts for an embedded processing system, a compiler is used to generate the appropriate object code from the source code that the programmer has written. In general, conventional compiler technology has evolved to allow

5   the user to select the quality of the code that is generated (e.g., compilers have compilation switches that allow either independent or linked optimization of the code for space and time). Generally, object code that is optimized for space (i.e., low memory requirements) runs slower than object code that has been optimized

10   for time (i.e., can use as much memory as necessary).

Circuit designers frequently make trade-offs when designing embedded processing systems. One major issue that must be resolved when designing an ASIC device or other embedded processing system is the amount of embedded memory that will be available in the

15   system. Because a memory circuit can be expensive in terms of space, power consumption, and speed, it is important to optimize the embedded memory to minimize these costs while retaining as much flexibility as possible.

Several different types of memories may be used in modern

20   embedded ASIC devices. These memories include SRAM, DRAM, flash RAM, EEPROM, flip-flops, and ROM. Each of these memories has different characteristics that make the memory more suitable or

less suitable for a particular application. Unfortunately, while

modern design tools, such as compilers and debuggers, are capable

of telling circuit designers the total amount of memory needed,

they tell very little else about the memory requirements of an ASIC

5   device. The software designers who write the code executed by the

ASIC device tend to treat all memory the same and do not write code

in a manner that exploits the characteristics of different types of

memories. The end result is that many ASIC devices run slower,

consume more power, or are larger in size than would otherwise have

10   been necessary.

Therefore, there is a need in the art for improved apparatuses

and methods for designing embedded processing systems. In

particular, there is a need for embedded processing system design

tools that are capable of determining the memory access usage of an

application executed by an embedded processing system that is under

15   design. More particularly, there is a need for embedded processing

system design tools that are capable of selecting and optimizing

the types and sizes of memories used in the target device and

optimizing the application program code executed by the target

20   device to exploit the characteristics of the different types of

available memories.

SUMMARY OF THE INVENTION

To address the above-discussed deficiencies of the prior art, it is a primary object of the present invention to provide an apparatus for designing and optimizing a memory for use in an embedded processing system. In an advantageous embodiment of the present invention, the apparatus comprises: 1) a simulation controller capable of simulating execution of a test program to be executed by the embedded processing system; 2) a memory access monitor capable of monitoring memory accesses to a simulated memory space during the simulated execution of the test program, wherein the memory access monitor is capable of generating memory usage statistical data associated with the monitored memory accesses; and 3) a memory optimization controller capable of comparing the memory usage statistical data and one or more predetermined design criteria associated with the embedded processing system and, in response to the comparison, determining at least one memory configuration capable of satisfying the one or more predetermined design criteria.

According to one embodiment of the present invention, the at least one memory configuration is determined from a predetermined set of memory types, the predetermined set of memory types

comprising at least two of static random access memory (SRAM),

dynamic random access memory (DRAM), read-only memory (ROM), flash

RAM (FLASH), and electronically erasable programmable read-only

memory (EEPROM).

5   According to another embodiment of the present invention, the

at least one memory configuration comprises a first memory type and

a first memory size associated with the first memory type.

According to still another embodiment of the present

invention, the at least one memory configuration further comprises

10   a second memory type and a second memory size associated with the

second memory type.

According to yet another embodiment of the present invention,

the simulation controller simulates execution of the test program

N times and wherein the memory access monitor monitors the memory

15   accesses during the N simulated executions of the test program and

generates the memory usage statistical data based on the N

simulated executions of the test program.

According to a further embodiment of the present invention,

the memory optimization controller is further capable of

20   determining at least one figure of merit associated with the at

least one memory configuration, wherein the at least one figure of

merit indicates a degree to which the at least one memory

configuration satisfies the one or more predetermined design criteria.

According to a still further embodiment of the present invention, the apparatus further comprises a code optimization controller capable of modifying the test program in response to the comparison of the memory usage statistical data and the one or more predetermined design criteria to thereby enable the embedded processing system to execute the test program according to the one or more predetermined design criteria.

The foregoing has outlined rather broadly the features and technical advantages of the present invention so that those skilled in the art may better understand the detailed description of the invention that follows. Additional features and advantages of the invention will be described hereinafter that form the subject of the claims of the invention. Those skilled in the art should appreciate that they may readily use the conception and the specific embodiment disclosed as a basis for modifying or designing other structures for carrying out the same purposes of the present invention. Those skilled in the art should also realize that such equivalent constructions do not depart from the spirit and scope of the invention in its broadest form.

Before undertaking the DETAILED DESCRIPTION OF THE INVENTION

below, it may be advantageous to set forth definitions of certain

words and phrases used throughout this patent document: the terms

"include" and "comprise," as well as derivatives thereof, mean

inclusion without limitation; the term "or," is inclusive, meaning

5   and/or; the phrases "associated with" and "associated therewith,"

as well as derivatives thereof, may mean to include, be included

within, interconnect with, contain, be contained within, connect to

or with, couple to or with, be communicable with, cooperate with,

interleave, juxtapose, be proximate to, be bound to or with, have,

10   have a property of, or the like; and the term "controller" means

any device, system or part thereof that controls at least one

operation, such a device may be implemented in hardware, firmware

or software, or some combination of at least two of the same. It

should be noted that the functionality associated with any

15   particular controller may be centralized or distributed, whether

locally or remotely. Definitions for certain words and phrases are

provided throughout this patent document, those of ordinary skill

in the art should understand that in many, if not most instances,

such definitions apply to prior, as well as future uses of such

20   defined words and phrases.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention, and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, wherein like numbers designate like objects, and in which:

FIGURE 1 illustrates an exemplary processing system capable of determining the memory requirements of an embedded system under design according to one embodiment of the present invention;

FIGURE 2 illustrates in greater detail memory design and optimization application programs that may be stored in the fixed disk drive and executed from the memory in FIGURE 1 according to one embodiment of the present invention; and

FIGURE 3 is a flow diagram illustrating the operation of the exemplary processing system according to one embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

FIGURES 1 through 3, discussed below, and the various embodiments used to describe the principles of the present invention in this patent document are by way of illustration only

5   and should not be construed in any way to limit the scope of the invention. Those skilled in the art will understand that the principles of the present invention may be implemented in any suitably arranged processing system.

FIGURE 1 illustrates exemplary processing system 100, which is

10   capable of designing and optimizing the memory of an embedded system, such as an application specific integrated circuit (ASIC), according to one embodiment of the present invention. In an advantageous embodiment, processing system 100 may be embodied in a personal computer (PC) or equivalent workstation (as shown in

15   FIGURE 1) that contains a processor and memory capable of executing memory design applications and/or memory optimization applications according the principles of the present invention.

Processing system 100 comprises data processor (CPU) 110, memory 120, removable media drive 130, fixed (i.e., "hard") disk

20   drive 140, user input/output (I/O) interface (IF) 150, keyboard 152, mouse 154 (or similar pointing device), video/audio

interface (IF) 160 and monitor 170.   Memory 120 may comprise

volatile memory, such as dynamic random access memory (DRAM), non-

volatile memory, such as flash RAM, or a combination of volatile

and non-volatile memory.   Removable media drive 130 may be any type

5   of storage device that is capable of reading from and/or writing to

a removable storage medium, such as a 3.5 inch floppy diskette, a

CD-ROM, a writable CD, a digital versatile disk (DVD), or the like.

A removable storage medium, such as CD-ROM 132, may be used to load

onto fixed disk 140 application programs and data, including the

10   memory optimization application programs explained below.   Fixed

disk drive 140 provides fast access for storage and retrieval of

application programs and data, including stored memory optimization

application programs according to the principles of the present

invention.

15   Keyboard 152 and mouse 154 are coupled to processing

system 100 via user I/O IF 150.   An embedded processing systems

designer uses keyboard 152 and mouse 154 to control the operation

of memory design/optimization applications embodying the principles

of the present invention and to enter data used by those

20   applications, such as user design criteria and memory models

(described below in greater detail).   Monitor 170 is coupled to

processing system 100 via video/audio IF 160.   The internal

components of processing system 100, including CPU 110, memory 120,

removable media drive 130, fixed disk drive 140, user I/O IF 150,

and video/audio IF 160 are coupled to and communicate across

internal communication bus 190.

5   In an advantageous embodiment of the present invention, a

memory design and optimization apparatus according to the

principles of the present invention may comprise a controller that

is implemented using a conventional data processor (i.e., CPU 110)

that executes one or more memory design and optimization

10 application programs stored in memory 120 and fixed disk drive 140.

Since the memory design and optimization application programs and

associated data files may be transferred into memory 120 from a

removable storage medium, the present invention may be implemented

as memory design and optimization application programs and

15 associated data files stored on, for example, CD-ROM 132.

        FIGURE 2 illustrates in greater detail memory design and

optimization application programs and data files that may be

executed from and stored in memory 120 (and stored in fixed disk

drive 140) according to one embodiment of the present invention.

20 The memory design and optimization application programs in

memory 120 are design tools that software and hardware designers

may use to determine the optimum memory requirements of an

exemplary ASIC device (occasionally referred to hereafter as the "target" device) that is being designed. Processing system 100 simulates the execution of the object code executed by the target device and then determines the types and amounts of memories that

5   may be used in the target device in order to meet certain user-specified design criteria.

Optionally, processing system 100 may determine several different memory configurations and may assign to each configuration one or more figures of merit (e.g., scale of 1 to 10

10   rating) that indicates how well each configuration meets the user-specified design criteria. In an advantageous embodiment of the present invention, processing system 100 may also be capable of modifying the object code executed by the target device in order to achieve an optimum solution of memory devices and software that

15   better meets the user-specified design criteria.

Memory 120 stores ASIC application source code file 205, ASIC application object code file 210, compiler program 215, instruction set simulator (ISS) program 220, simulated ASIC memory space 230, debugger program 235, code optimizer program 240, histogram

20   file 250, memory models file 260, and user design criteria file 270, and memory design and optimization program 280. It should be noted that the exemplary programs depicted in memory 120

reflect only one possible logical division of the functions of a memory design tool according to the principles of the present invention. In reality, all or parts of one or more of the exemplary programs may be combined into other programs. For example, compiler program 215 may actually be a "smart" compiler program containing sub-routines that incorporate one or more of instruction set simulator (ISS) program 220, debugger program 235, code optimizer program 240, and memory design and optimization program 280.

ASIC application source code file 205 comprises the proposed source code written by the software designers to operate the target device under design. Processing system 100 executes compiler program 215 in order to compile the source code and produce executable object code that is stored in ASIC application object code file 210.

When the object code is compiled, the system designer may then use processing system 100 to run instruction set simulator (ISS) program 220 on the compiled object code. ISS program 220 simulates the execution of the compiled object code by the target device in simulated ASIC memory space 230. The object code itself is copied into simulated ASIC memory space 230 and all memory access operations (i.e., read operations and write operations) occur

within the memory space of simulated ASIC memory space 230. As the

execution of the object code is simulated, debugger 235 is capable

of working with ISS program 220 to permit the designer to track the

simulated execution of the object code.

5   As the execution of the object code is repeatedly simulated,

ISS program 220 monitors all memory access operations and creates

in histogram file 250 a plurality of histograms of all memory

access operations. These histograms may include, among others,

variables histogram file 252, which comprises one or more

10   histograms based on variable names contained in the object code

executed by the target device under design, and memory location

histogram file 254, which comprises one or more histograms based on

memory locations accessed by the object code executed by the target

device.

15   When an initial histogram of the object code has been

prepared, the system designer may then use processing system 100 to

run memory design and optimization program 280. Memory design and

optimization program 280 uses the data in histogram 250, the data

in memory models file 260, and the data in user design criteria

20   file 270 to determine the types and amounts of memory that should

be used in the target device to best meet the operating parameters

specified by the user in user design criteria file 270. In one

embodiment of the present invention, the data in user design criteria file 270 may specify general objectives for the target device (i.e., minimize SRAM usage, maximize ROM usage, minimize power consumption, and the like). In alternate embodiments of the present invention, the data in user design criteria file 270 may specify more quantitative objectives for the target device, such as a maximum of N kilobits of SRAM, a maximum of R watts of power consumption, a maximum write operation access speed, and the like.

Table 1 and Table 2 below illustrate exemplary contents of memory models file 260. The data in memory models file 260 specifies the relative performance advantages and disadvantages of a plurality of memory types, including static random access memory (SRAM), dynamic random access memory (DRAM), read-only memory (ROM), flash RAM (FLASH), and electronically erasable programmable read only memory (EEPROM).

Memory Type	Write Power	Refresh Power	Read Power	Area Per bit	Write Speed
SRAM	high	n/a	mid	high	fast
DRAM	low	high	mid	low	fast
ROM	n/a	n/a	low	low	n/a
FLASH	high	n/a	mid	low	slow
EEPROM	low	n/a	mid	mid	mid

TABLE 1

Memory Type	Read Speed	Erase Capability	Block Size	Area Efficiency
SRAM	fast	yes	all	high
DRAM	fast	yes	all	OK
ROM	fast	no	all	good
FLASH	mid	yes	limited	OK
EEPROM	mid	yes	limited	OK

TABLE 2

Given an application to be executed by the target device, memory design and optimization program 280 and code optimizer 240 help the designer select memory sizes and types given the constraints in user design criteria file 270.

In response to the memory configurations and/or figures of merit determined by memory design and optimization program 280, code optimizer 240 may re-order and/or re-write selected portions of the compiled object code in order to achieve greater efficiencies and to better meet the constraints specified in user design criteria file 270. For instance, code optimizer 240 and/or memory design and optimization program 280 can modify the object code to store one or more sparsely used variables into an address space that corresponds to a flash memory that is cheaper that SRAM in terms of cell area, but slower in terms of write speed. Also, a variable name that counts errors and that is very infrequently used may be re-written by code optimizer program 240 so that

successive writes to the variable can be stored in consecutive memory locations (as opposed to the same one as a standard compiler would so).

Table 3 and Table 4 below illustrate an additional example of a portion of code that has been re-written by code optimizer program 240 to operate in a more efficient manner. For the original code in Table 3, it is assumed that the variable J is changed in the outer loop and is continually read and written from a conventional SRAM.

```
        For  (J=0;  J<N;  J++)
    {
        For  (I=0;  I<M;  I++)
        {
            [BLOCK OF EXECUTABLE CODE]
        }
    }
```

TABLE 3

Code optimizer program 240 creates the new code in Table 4, which makes J into an array in flash RAM that is written and read in consecutive locations:

```
For (J[mem_access=0]=0; J[mem_access]<N; J[mem_access+1]=
J[mem_access]+1, mem_access++)
{
        For     (I[mem_access1=0]=0;      I[mem_access1]      <N;
I[mem_access1+1] = I[mem_access1]+1, mem_access1++)
        {
                [BLOCK OF EXECUTABLE CODE]
        }
}
```

                                TABLE 4

Mem_access and mem_access1 are now stored in the memory
controller block, not in SRAM.

Processing systems 100 may also use a software (or hardware)
run-time memory manager which can redirect memory accesses to
better memory that is available. In the hardware implementation,
memory optimization would use a memory interface controller
designed to track memory accesses and to reroute the accesses to
better utilized blocks of memory to optimize some power or speed
constraints. For instance, if an address is not frequently used
(as determined by a LRU algorithm), then the memory manager can
copy less frequently used data to slower memory with lower power
constraints (e.g., DRAM and SRAM mapped variables may instead be
stored in flash RAM).

FIGURE 3 depicts flow diagram 300, which illustrates the
operation of exemplary processing system 100 according to one
embodiment of the present invention. Initially, processing

system 100 compiles a source code file prepared by the embedded

system designer to produce an object code file (process step 305).

Then, processing system 100 then executes ISS program 220 to

simulated the execution of the compiled object code (process

step 310). Processing system 100 also runs debugger program 235 in

order to debug and edit the object code, if necessary, as it runs

(process step 315). As it is executed, ISS program 220 monitors

the memory accesses in simulated ASIC memory space 230 and gathers

memory usage statistics (process step 320). Simultaneously, ISS

program 220 creates or updates the memory access histograms in

histogram file 250 (process step 325).

Optionally, processing system 100 may execute code optimizer

program 240 in order to modify the object code in response to the

histogram data (process step 330). Processing system 100 continues

to loop through process steps 310, 315, 320, 325 and 330 until a

sufficient number of loops have been performed to ensure that the

data in histogram file 250 is an accurate reflection of the real-

world memory access usage of the embedded ASIC device being

designed. The number of loops may be user-defined or execution of

the loop may terminate when the histogram data converges to a

reasonably stable value (process step 335).

When a sufficient number of loops have been performed, memory

design and optimization program 280 analyzes the histogram data and the user-specified values in user design criteria file 270 and determines one or more memory configurations for the target device that satisfy the user design criteria. Optionally, memory design

5   and optimization program 280 may determine one or more figures of merit for each memory configuration to indicate how well the memory configuration satisfies the user design criteria (process step 340). At this point, processing system 100 may (optionally) execute code optimizer program 240 in order to modify the object

10   code in response to the histogram data, the selected memory configuration, and the figures of merit (process step 345).

Thereafter, processing system 100 may loop back to process step 310 until a sufficient number of loops have been performed to ensure that the data in histogram file 250 is an accurate

15   reflection of the real-world memory access usage of the selected memory configuration. Again, the number of loops may be user-defined or execution of the loop may terminate when the user design criteria are adequately met by one or more of the memory configurations determined by processing system 10 (process

20   step 350).

Although the present invention has been described in detail, those skilled in the art should understand that they can make

various changes, substitutions and alterations herein without

departing from the spirit and scope of the invention in its

broadest form.